Manual de usuario de la Intel simplificada

Ing. Donald Velásquez.
Ing. Cristian Peláez.
Ing. Jonathan de los Santos.

Índice de contenido

Prefacio....2
La forma correcta de programar....3
La interfaz del programa....3
Un vistazo rápido a todas las instrucciones....4
La instrucción ADD....4
La instrucción NOT....5
La instrucción AND....5
La instrucción NEG....6
La instrucción OR....6
La instrucción XOR....7
La instrucción JMP....8
La instrucción HLT....8
La instrucción IN....8
La instrucción MOV....9
La instrucción Jxxx....9
La instrucción MLT....10
La instrucción LEA....11
La instrucción LDS....11
La instrucción DIV....11

I. Prefacio

Este pequeño manual de usuario se ha elaborado con el fin de poder orientar al a lector sobre la manera de cómo se programa correctamente el proyecto de la Intel simplificada. Antes que nada hay que saber que el proyecto es una verdadera simplificación de lo que es en realidad la máquina Intel que posee aproximadamente 140 instrucciones para su vital funcionamiento. En este proyecto únicamente se implementaron 16 instrucciones vitales para propósitos generales que su elección se basó en la experiencia de los creadores así como en la máquina LC-3, que como su nombre lo indica es una “Little Computer” por sus siglas en inglés.

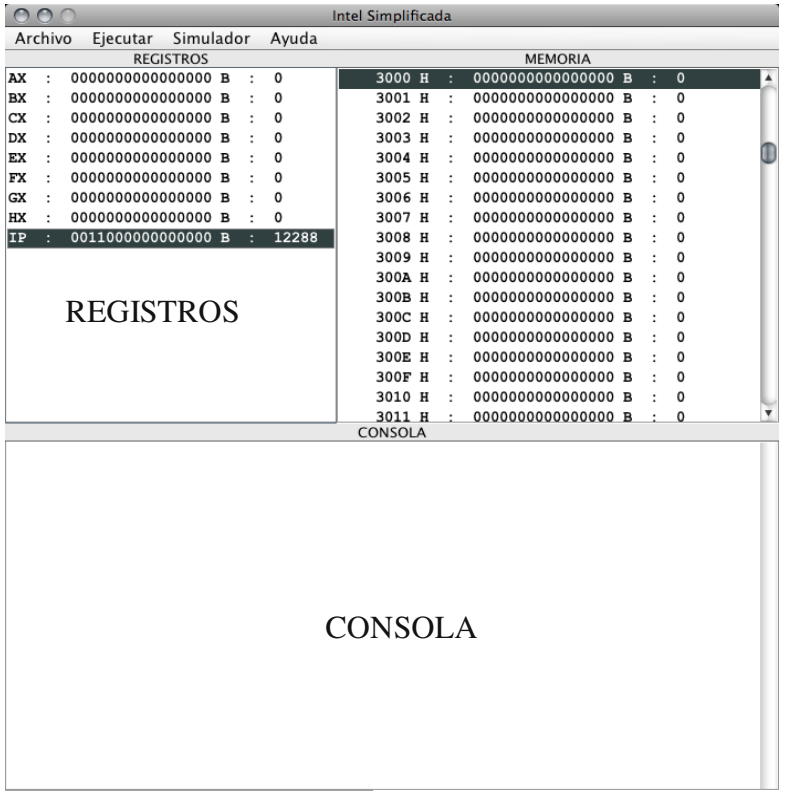
A pesar de su parecido con la LC-3 nuestro proyecto contiene algunas instrucciones que facilitan al programador implementar funciones aritméticas como multiplicación, o negación a complemento a dos, desde otro punto de vista nuestro proyecto puede ser una nueva versión más optimizada de la LC-3. A continuación se explica con detalle cada una de las normas del programa ensamblador de nuestro proyecto, son reglas bastante sencillas pero es importante seguirlas para el buen funcionamiento del traductor.

II. La forma correcta de programar

Como todo lenguaje de programación la forma de programar varía como varía el número de máquinas en el mundo, nuestro lenguaje ensamblador es bastante sencillo y muy parecido a otros aunque no igual. Se recomienda antes que nada que a la hora de programar siga una estructura ordenada dejando las tabulaciones adecuadas para cada bloque de instrucciones. Lo primero que tiene que hacer cuando empieza un programa es declarar las instrucciones ORG y END al principio y al final respectivamente de todo el código. Estas establecen donde comienza y donde terminar el conjunto de instrucciones, para hacer comentarios utilice ";" antes de cada linea a comentar, comentar bien sus programas le ayudará a encontrar más rápido sus errores. Para saber la sintaxis de cada una de las instrucciones no hace falta más que se refiera a la siguiente tabla, los OPCODE son los números que distinguen a cada instrucción de otra, los DST son declaraciones de registros en donde se va a guardar un resultado, los SRC son los registros de donde obtendremos un operando, los valores inmediatos y los PCOffset funcionan similarmente. Para saber la función de cada una de las instrucciones refiérase a su sección individual contenida en el manual, aquí se especifica con lujo de detalles lo que es capaz de hacer cada instrucción.

III. La interfaz del programa

MEMORIA



Este es sólo un preámbulo a lo que se encontrará al iniciar el programa, puede ver que es muy amigable y en sus menús podrá encontrar todo lo que necesita para programar y revisar su código paso por paso para encontrar posibles errores.

1966 UNIVERSIDAD DEL VALLE DE GUATEMALA
Excelencia que trasciende

IV. Un vistazo rápido a todas las instrucciones

INST.	**OPCODE**				**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADD	0	0	0	0	DST			SRC1			0	0	0	SRC2		
ADD	0	0	0	0	DST			SRC			1	Inmediato				
NOT	0	0	0	1	DST			SRC			1	1	1	1	1	1
AND	0	0	1	0	DST			SRC1			0	0	0	SRC2		
AND	0	0	1	0	DST			SRC			1	Inmediato				
NEG	0	0	1	1	DST			SRC			1	1	1	1	1	1
OR	0	1	0	0	DST			SRC1			0	0	0	SRC2		
OR	0	1	0	0	DST			SRC			1	Inmediato				
XOR	0	1	0	1	DST			SRC1			0	0	0	SRC2		
XOR	0	1	0	1	DST			SRC			1	Inmediato				
JMP	0	1	1	0	0	0	0	SRC			0	0	0	0	0	0
HLT	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
IN	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
MOV	1	0	0	1	DST			SRC			1	1	1	1	1	1
J	1	0	1	0	n	z	p	PCOffset9								
MLT	1	0	1	1	DST			SRC1			0	0	0	SRC2		
MLT	1	0	1	1	DST			SRC			1	Inmediato				
LEA	1	1	0	0	DST			PCOffset9								
LDS	1	1	0	1	SRC			PCOffset9								
DIV	1	1	1	0	DST			SRC1			0	0	0	SRC2		
DIV	1	1	1	0	DST			SRC			1	Inmediato				

V. La instrucción ADD

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
ADD	0	0	0	0	DST			SRC1			0	0	0	SRC2		
ADD	0	0	0	0	DST			SRC			1	Inmediato				

Como podemos ver en la sintaxis de la instrucción ADD se pueden tener dos modos de operación especificados por el valor del bit 5, este bit con el valor de cero especifica el modo de operación entre dos registros, desde ahora en adelante llamaremos este modo de operación Registro-Registro, y el otro modo de operación es cuando el bit 5 tiene el valor de uno, que se usa para especificar el modo de operación Registro-Inmediato. El valor inmediato es un número puro que no se encuentra dentro de un registro ni dentro de una localidad de memoria si no que el programador lo escribe según su conveniencia en su código fuente.

Ejemplo de ADD con modo de operación Registro-Registro.

Assembly	**Binario**
ADD AX,DX,EX	0000000011000100

Suponga que en el registro DX está almacenado el valor FH, y que en EX el valor AH entonces después de pasar esta

1966 UVG Universidad del Valle de Guatemala
Excelencia que trasciende

instrucción el valor de AX será (FH+AH)=19H=25.

Ejemplo de ADD con modo de operación Registro-Inmediato.

Assembly	Binario
ADD AX,DX,15	0000000011101111

Suponga que en el registro DX está almacenado el valor FH, entonces el valor de AX después de la ejecución de esta instrucción será (FH+15)*=(FH+FH)=1EH=30

*Note que el traductor es capaz de distinguir entre las bases numéricas Hexagesimal, Binario y Decimal.

VI. La instrucción NOT

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
NOT	0	0	0	1	DST			SRC			1	1	1	1	1	1

La instrucción NOT es una de las más simples en su sintaxis, únicamente en los bits 11-9 se especifica el registro destino, y en los bits 8-6 se especifica la fuente, la instrucción lo único que hace es cambiar los bits del dato contenido en la fuente uno por uno. El ejemplo se hace en binario para mayor comprensión.

Ejemplo de ADD con modo de operación Registro-Inmediato.

Assembly	Binario
NOT FX,DX	0001101011111111

Ahora suponga que en en el registro DX está almacenado el valor 1110001110001110B entonces el registro FX se cargará con el valor 0001110001110001B.

VII. La instrucción AND

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
AND	0	0	1	0	DST			SRC1			0	0	0	SRC2		
AND	0	0	1	0	DST			SRC			1	Inmediato				

La instrucción AND es lo que su nombre y teoría indica, es una simple operación lógica bit por bit y también puede trabajar con ambos modos de operación el registro-registro y el registro-inmediato.

Ejemplo de AND con modo de operación Registro-Registro.

Assembly	Binario
AND AX,DX,EX	0000000011000100

1966 UVG Universidad del Valle de Guatemala
Excelencia que trasciende

Suponga que en el registro DX está almacenado el valor FH, y que en EX el valor AH entonces después de pasar esta instrucción el valor de AX será:

0000000000001111B
0000000000001010B

0000000000001010B=AX

Ejemplo de AND con modo de operación Registro-Inmediato.

Assembly	Binario
AND AX,DX,15	0000000011101111

Una idea parecida es usando el modo de operación registro-inmediato, suponga que en el registro DX está almacenado el valor EH, entonces el valor de AX después de la ejecución de esta instrucción será:

0000000000001110B
0000000000001111B

0000000000001110B=AX

VIII. La instrucción NEG

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
NEG	0	0	1	1	DST			SRC			1	1	1	1	1	1

La función de la instrucción NEG es tomar el valor de la fuente especificada en los bits 8-6 y guardarla en el destino especificado en los bits 11-9 como su negativo en complemento a dos. El ejemplo se hace en base decimal para su compresión.

Assembly	Binario
NEG FX,DX	0011101011111111

Suponga que en el registro DX se almacena el valor 3454 entonces en FX se almacenará el valor -3454.

IX. La instrucción OR

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
OR	0	1	0	0	DST			SRC1			0	0	0	SRC2		
OR	0	1	0	0	DST			SRC			1	Inmediato				

Esta instrucción tiene la misma idea que la AND, solo que lógicamente hace la operación OR.

Ejemplo de OR con modo de operación Registro- Registro.

Assembly	Binario
OR AX,DX,EX	0000000011000100

Suponga que en el registro DX está almacenado el valor FH, y que en EX el valor AH entonces después de pasar esta instrucción el valor de AX será:

0000000000001111B
0000000000001010B

0000000000011001B=AX

Ejemplo de OR con modo de operación Registro-Inmediato.

Assembly	Binario
OR AX,DX,15	0000000011101111

De igual forma en en su modo de operación Registro inmediato, suponga que DX tiene almacenado el valor 14

0000000000001110B
0000000000001111B

0000000000011100B=AX

X. La instrucción XOR

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
XOR	0	1	0	1	DST			SRC1			0	0	0	SRC2		
XOR	0	1	0	1	DST			SRC			1	Inmediato				

La operación lógica XOR indica que para dos bits iguales el resultado es cero y para bits diferentes el resultado es uno, también posee ambos modos de operación.

Ejemplo de XOR con modo de operación Registro-Registro.

Assembly	Binario
XOR AX,DX,EX	0000000011000100

Suponga que en el registro DX está almacenado el valor AAH, y que en EX el valor AH entonces después de pasar esta instrucción el valor de AX será:

0000000010101010B
0000000000001010B

0000000010100000B=AX

Ejemplo de XOR con modo de operación Registro-Inmediato.

Assembly	Binario
XOR AX,DX,15	0000000011101111

De igual forma en en su modo de operación Registro-inmediato, suponga que DX tiene almacenado el valor CCH

0000000011001100B
0000000000001111B

0000000011000011B=AX

XI. La instrucción JMP

Esta instrucción es muy conocida en el lenguaje ensamblador ya que es de gran utilidad para bifurcaciones, nuestra forma de programar especialmente se basa en el valor contenido en el registro fuente, el IP (pointer instrucción) se cambia a este valor y el programa se seguirá ejecutando en esta dirección.

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
JMP	0	1	1	0	0	0	0	SRC			0	0	0	0	0	0

Por ejemplo la siguiente instrucción suponiendo que PC apunta a 3030H y HX tiene un valor de 3200H.

Assembly	Binario
JMP HX	0110000111000000

Después de la ejecución de esta instrucción el PC tendrá el valor 3200H y ejecutará la instrucción que está almacenada en dicha dirección, luego seguirá avanzando de uno en uno como antes.

XII. La instrucción HLT

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
HLT	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

La instrucción HALT es una de las más fáciles de entender ya que sólo tiene una forma única y sirve para detener el proceso de nuestro programa. Su forma estándar es:

Assembly	Binario
HLT	0111111111111111

Y después de esto el programa se detiene completamente mostrando la notificación en la consola.

XIII. La instrucción IN

NST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
IN	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

1966 UNIVERSIDAD DEL VALLE DE GUATEMALA
Excelencia que trasciende

Esta instrucción también tiene una única forma al igual que HLT a diferencia esta despliega un mensaje en la consola llamado prompt que pide al usuario que presione una tecla, al momento que se presiona esta se despliega en pantalla, a esta acción se le llama ECO.

Assembly	Binario
IN	1000111111111111

Después de esto el programa muestra el promt en la consola y se espera a la pulsación de la tecla por parte del usuario.

XIV. La instrucción MOV

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
MOV	1	0	0	1	DST			SRC			1	1	1	1	1	1

La instrucción MOV es una de las instrucciones más importantes de nuestro proyecto, se trata de poder almacenar lo que se encuentra en una localidad de memoria en un registro, la localidad de memoria se especifica con el valor almacenado en el registro SRC, se va a esa localidad de memoria, se haya su contenido y este se guarda en DST. Por ejemplo la instrucción:

Assembly	Binario
MOV DX,AX	1001011000111111

Suponga que en AX está el valor 4430H, este valor es el que especifica la localidad de memoria a la cual se accederá. Suponga que en la localidad de memoria 4430H se almacena el valor 110001B, este último valor es el que se almacenará en DX.

DX=110001B

XV. La instrucción Jxxx

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
J	1	0	1	0	n	z	p	PCOffset9								

Esta instrucción también es muy importante y sirve para hacer bifurcaciones y modificar el rumbo de nuestro IP, los bits 11-9 corresponden a las banderas nzp respectivamente, estas banderas se levantan dependiendo del resultado e la última instrucción realizada, si el resultado fue positivos la bandera p se pone en 1, si el resultado fue negativo la bandera n es la que se pone en 1, si fue cero en ese caso la bandera z se pone en 1. si alguna de las instrucciones se cumple según lo especificado se hace un salto del número de instrucciones especificadas en los bits 8-0, si es negativo se le resta al IP aumentado lo que es un salto hacia arriba si es positivo se le suma al IP aumentado lo que es un salto hacia abajo. Suponga en el siguiente ejemplo que IP=3030 y que el último valor recibido fue un positivo.

Assembly	Binario
Jp -2	1010001111111110

Por lo que ahora el IP vale 3029, si el resultado anterior no cumpliera con ser positivo la instrucción anterior se ignora y se pasa a la siguiente. Tome en cuenta los siguientes casos siguiendo la misma idea de lo explicado.

Instrucción	Descripción
Jn	Si se quiere evaluar un negativo
Jz	Si se quiere evaluar un cero
Jp	Si se quiere evaluar un positivo
Jnz	Cualquiera no positivo
Jnp	Cualquiera diferente de cero
Jzp	Cuaquiera no negativo
Jnzp	Todos los casos

XVI. La instrucción MLT

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
MLT	1	0	1	1	DST			SRC1			0	0	0	SRC2		
MLT	1	0	1	1	DST			SRC			1	Inmediato				

La instrucción MLT es una multiplicación aritmética entre los valores de sus operandos, importante no confundir la con AND, AND es una operación lógica y no aritmética. Vea su diferencia en los siguientes ejemplos.

Ejemplo de MLT con modo de operación Registro-Registro.

Assembly	Binario
MLT AX,DX,HX	1011000011000111

Suponga que el valor de DX es 1000 y el de HX es 5, entonces AX=(DX)(HX)=5000, note que se operan los valores contenidos en los registros y que el resultado no puede ser mayor a 65535 ya que se exceden los 16 bits que tiene cada registro.

Ejemplo de MLT con modo de operación Registro-Inmediato.

Assembly	Binario
MLT AX,DX,FH	1011000011000111

Suponga que el valor de DX es 1000 entonces AX=(1000)(15)=15000

XVII. La instrucción LEA

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
LEA	1	1	0	0	DST			PCOffset9								

La instrucción LEA se refiere a un modo inmediato de almacenamiento de LOCALIDADES DE MEMORIA A UN REGISTRO, se basa en el IP aumentado. DST es el registro donde se va a guardar la dirección apuntado al sumar algebraicamente el IP aumentado con el valor indicado por los bits 8-0 en complemento a dos. Para el siguiente ejemplo suponga que el IP apunta a la localidad 3232H

Assembly	Binario
LEA CX,255	110001001111111

Entonces CX=3232H+1+FFH=3332H, note que en ningún momento se afecta al IP, solamente se usa su valor aumentado en uno.

XVIII. La instrucción LDS

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
LDS	1	1	0	1	SRC			PCOffset9								

El LDS funciona de manera similar al MOV, sin embargo LDS sirve para escribir valores en la memoria. Trabaja también con el IP aumentado y la dirección de la localidad de memoria a escribir está dada por la suma del IP aumentado sumado algebraicamente con el PCOffset especificado, el valor que se escribe en tal localidad de memoria es el contenido en el registro especificado en los bits11-9. Por ejemplo suponga que IP vale 16798 y que en el registro CX está almacenado el valor 2000.

Assembly	Binario
LDS CX,200	1101010011001000

Al finalizar esta instrucción la localidad de memoria 16999 tendrá el valor 2000.

XIX. La instrucción DIV

INST.	OPCODE				11	10	9	8	7	6	5	4	3	2	1	0
DIV	1	1	1	0	DST			SRC1			0	0	0	SRC2		
DIV	1	1	1	0	DST			SRC			1	Inmediato				

DIV es una instrucción que hace una división aritmética en donde no se toma en cuenta ningún valor en decimales, es decir únicamente los enteros de dichos números. Su modo de operación es similar a la de la instrucción MLT.

1966 UVG UNIVERSIDAD DEL VALLE DE GUATEMALA
Excelencia que trasciende

Ejemplo de DIV con modo de operación Registro-Registro.

Assembly	Binario
DIV AX,DX,HX	1011000011000111

Suponga que el valor de DX es 1000 y el de HX es 5, entonces AX=(DX)/(HX)=200, note que se operan los valores contenidos en los registros y que el resultado no puede ser mayor a 65535 ya que se exceden los 16 bits que tiene cada registro, esto sólo sucedería si HX fuera un número entre 0 y 1 pero no tenemos de que preocuparnos porque esto, ni la división dentro de cero va a suceder, el programa notifica del error al programador.

Ejemplo de DIV con modo de operación Registro-Inmediato.

Assembly	Binario
MLT AX,DX,FH	1011000011000111

Suponga que el valor de DX es 1000 entonces AX=(1000)/(15)=66